Custom JSON-Driven SFTP Protocol

Documentation

Author: Trey Rubino

Table of Contents

1. Request Class

The Request class represents client-side requests in the protocol. It inherits from CustomProtocol.

1.1 Attributes and Data Types

Attribute	Type	Description
cmd	String	The command to be executed (e.g., 'ls', 'cd', 'get').
local_path	Optional[String]	The path on the local machine.
remote_path	Optional[String]	The path on the remote server.
recursive	Optional[Boolean]	Whether the command applies recursively.
size	Optional[Integer]	Size of the file for upload/download.

1.2 Examples of Valid Payloads

```
{
  "cmd": "ls",
  "local_path": "/home/user/",
  "recursive": false
}



{
  "cmd": "put",
  "local_path": "/home/user/file.txt",
  "remote_path": "/server/uploads/file.txt",
  "size": 1024
}
```

2. Response Class

The Response class represents server responses in the protocol. It inherits from CustomProtocol.

2.1 Attributes and Data Types

Attribute	Type	Description
status	String	The response status (e.g., 'success', 'error').
message	Optional[String]	A descriptive message accompanying the response.
contents	Optional[List[Content]]	List of directory or file entries for the 'ls' command.
code	Optional[String]	Error or status code for troubleshooting.

2.2 Examples of Valid Payloads

```
{
   "status": "success",
   "message": "Directory contents listed successfully.",
   "contents": [
      {
         "name": "file1.txt",
         "size": 1024,
         "type": "file"
      },
      {
         "name": "subdir",
         "size": 0,
         "type": "dir"
      }
   ]
}



{
   "status": "error",
   "message": "Invalid directory path.",
   "code": "ERR_INVALID_DIR"
}
```

3. CustomProtocol Class

The CustomProtocol class provides shared functionality for handling JSON encoding/decoding and binary data.

3.1 Methods and Descriptions

Method	Description
validate()	Ensures the object meets required criteria.
prepare()	Validates and encodes the object into JSON bytes.
encode()	Encodes the object as JSON bytes.
decode(data, cls)	Decodes JSON bytes into an instance of the specified class.
attach_binary_data(data)	Attaches binary data to the object.
get_binary_data()	Retrieves attached binary data.

4. Utility Class

The Utility class provides methods for handling various commands (e.g., ls, cd, mkdir, get, put) and facilitates sending and receiving data between the client and server.

4.1 Methods and Descriptions

Method	Description
ls(request)	Lists directory contents.
cd(request)	Changes the current working directory.
mkdir(request)	Creates a new directory.
get(conn, request)	Downloads a file from the server.
put(conn, request)	Uploads a file to the server.
send_all(conn, obj)	Sends JSON and binary data to the specified connection.
recv_all(conn, obj_type)	Receives JSON and binary data from the specified connection.

5. Error Handling

Error handling is a critical part of the protocol, ensuring robust communication and clear feedback for errors encountered during operation.

5.1 Common Error Codes and Explanations

Error Code	Description
ERR_INVALID_DIR	The specified directory path is invalid.
ERR_DIR_NOT_FOUND	The requested directory could not be found.
ERR_PERMISSION_DENIED	Insufficient permissions to access the specified path.
ERR_DIR_EXISTS	The directory already exists.
ERR_GET_CLIENT	An error occurred while retrieving a file from the server.
ERR_PUT_CLIENT	An error occurred while uploading a file to the server.
ERR_CONNECTION_LOST	The connection was lost during data transfer.